Amendments to the Specification:

Please replace the paragraph beginning at page 8, line 12, with the following rewritten paragraph:

2. A secondary allocation will be taken and used on the volume in which the initial close occurred following a database load. For example, in the case of a first database, ~~Database1~~database1, the close occurs on the last volume as determined by a ~~listcat~~LISTCAT. Secondary allocations will only occur on that last volume after all of the primary is used on the other volumes. In the case of a second database, database2, the close occurs on the second volume. Secondary allocations will be taken and used before the last primary is used even though it is already allocated.

Please replace the paragraph beginning at page 9, line 15, with the following rewritten paragraph:

SUBLISTC 300 (Fig. 3) – issues an IDCAMS LISTCAT command for a dataset that was passed to it, and puts the results in a dataset.

SUB009 400 (Figs. 4A-4C) – Reads the output from a SUBLISTC for a dataset and returns the following information – ~~gts~~ gts_flag, last volume, total number of volumes and 20 occurrences of volume serial numbers for a dataset.

SUB006 500 (Figs. 5A-5C) – Reads the output from a SUBLISTC and returns the following information – ~~gts~~ gts_flag and last volume.

IEHLISTR 600 (Fig. 6) – Dynamically executes an IEHLIST utility.

SUBL002 700 (Figs. 7A and 7B) – Reads the output from the IEHLISTR subroutine and returns the following information – total free cylinders.

Please replace the paragraph beginning at page 10, line 15, with the following rewritten paragraph:

Returning to the ~~**PROCESS_FIRST_TIME**~~process_first_time step 212, first all variables are initialized in an initialize all variables step 213. Next, total variables are initialized in an initialize total variables step 215, and then process 200 proceeds to ~~Process~~process_same_dataset step 208. The ~~**PROCESS_NEW_DATASET**~~ process_new_dataset step 214 is followed by a ~~Write~~write_out_dataset step 220. All variables are initialized in an initialize all variables step 222, which is in turn followed by a ~~Get~~get_listdsi step 224. Next, a ~~Determine~~determine_total_num_vols operation is performed in step 226, which is followed by the ~~Process~~process_same_dataset operation in step 208.

Please replace the paragraph beginning at page 10, line 22, with the following rewritten paragraph:

The ~~**PROCESS_SAME_DATASET**~~process_same_dataset step 208 adds one to an index and puts the variables from step 204 into a current entry list which has a maximum of 20 entries.

Please replace the paragraph beginning at page 10, line 24, with the following rewritten paragraph:

The ~~WRITE_OUT_DATASET_INFO~~write_out_dataset_info step ~~226~~220 begins with calling the SUBLISTC subroutine 300 ~~subliste~~ SUBLISTC using dataset name. Subroutine 300 in turn proceeds to call subroutine 400 ~~sub009~~SUB009. These subroutines are described in further detail below in connection with the discussions of Figs. 3 and 4A-4C, respectively. A get step 221 gts_flag and all_volsers from ~~sub009~~ SUB009 400. The volsers are split in a ~~Splits~~split_volsers step 222 and then each item is processed serially in step 225. As used here, serial processing means that decision blocks 227, 229, and 231 are all processed serially, regardless of the answer to a preceding block. In step 227, it is determined if the variable used is all zeroes, if yes, then space is adjusted in an adjust_space step 228. If as part of serial processing step 225, it is determined in step 229 that if all zeroes and the volsers are out of sync, then ordinates are retrieved in a get_ordinates step 230. Alternatively, if an all zeroes determination is made at step 231, then a ~~Total~~total_up_alloc step 232, followed by a ~~Total~~total_up_used step 234, and a ~~Calc~~calc_percent_used step 236 result. Following serial processing step 225, a write blank line step 238 occurs. In step 240, if gts_flag = "YES", then a display or write message step 241 results. If the gts_flag = "NO", it is determined in step 242 if the number of candidates is greater than 90, then the display_message step 241 again results. If the number of candidates is not greater than 90 in step 242, the process proceeds to step 244 to calculate the number of volumes left based on data class, before proceeding to the write message step 241.

Please replace the paragraph beginning at page 11, line 17, with the following rewritten paragraph:

If it is determined in step 246 that the volumes are out of sync, then display message step 247 displays a message to this effect. If it is determined that the volumes are not out of sync in step 248, then a message to this effect is displayed in step 249. In step 250, it is determined whether there is space used. More particularly, if all zeroes is not = zeroes, there is space used. If space is used, then ~~Further~~further_process_output step 251 proceeds. If not, a display message step 252 is followed by a ~~Determine~~determine_over_threshold step 254.

Please replace the paragraph beginning at page 11, line 23, with the following rewritten paragraph:

Returning to ~~**FURTHER_PROCESS_OUTPUT**~~further_process_output step 251, a report line is formatted for each volser, and then these lines are written.

Please replace the paragraph beginning at page 12, line 1, with the following rewritten paragraph:

Returning to ~~**SPLIT_VOLSERS**~~split_volsers step 223, all volsers from a subroutine are parsed to listcat_volser_tables, and a number_candidates step 233 determines the number of entries. If there are more than 20 entries, in other words, if there are more than 20 volumes, there is a more serious problem. In step 235, a write message provides an indication of this severe error as follows.

Please replace the paragraph beginning at page 12, line 6, with the following rewritten paragraph:

~~**ADJUST_SPACE**~~Adjust_space step 228 is followed by a decision step 237. If there are more than 20 volsers, then a message is played in response to write error step and the exit code is set to 16 indicative of a severe error. This process looks at the information from the ~~listcat~~ LISTCAT and the ~~dcollect~~ DCOLLECT to make sure that the order is correct. It sets the used equal to the alloc based on a switch (found_end_vol). This switch is normally set off, and turned on when a non-zero value is found. If the found_end_vol switch is on, then the adjustment is ignored. This is because when there is a multi-volume OSAM dataset in IMS, the volumes prior to the initial close are set to null, and show zero, even if there is space used.

Please replace the paragraph beginning at page 12, line 14, with the following rewritten paragraph:

Returning to ~~**TOTAL_UP_ALLOC**~~total_up_alloc step 232, this step adds alloc_space for each_volser. Similarly, ~~**TOTAL_UP_USED**~~total_up_used step 234 adds used_space for each_volser. ~~**CALC_PERCENT_USED**~~ Calc_percent_used step 236 calculates if the used_space and the alloc_space are not equal to zero, then the percent_used for volume is calculated.

Please replace the paragraph beginning at page 12, line 18, with the following rewritten paragraph:

With respect to ~~**GET_ORDINATES**~~get_ordinates step 230, it is determined if the volser entry = listcat_volser, by matching ~~listcat~~LISTCAT to ~~dcollect~~DCOLLECT. Next, the order number is put in the order variable for the row.

Please replace the paragraph beginning at page 12, line 21, with the following rewritten paragraph:

~~**GET_LISTDSI**~~Get_listdsi step 224 gets dataset information for directory, no-recall and smsinfo. LISTDSI is a standard REXX routine to determine dataset attributes.

Please replace the paragraph beginning at page 12, line 23, with the following rewritten paragraph:

~~**DETERMINE_TOTAL_NUM_VOLS**~~Determine_total_num_vols step ~~256~~226 strips out the fourth and fifth position from the data class and uses them as the number of volumes. This approach is a standard approach to determining the total number of volumes.

Please replace the paragraph beginning at page 13, line 1, with the following rewritten paragraph:

~~**WRITE_OUTFILE1**~~Write_outfile1 step 258 operates to write an error report.

Please replace the paragraph beginning at page 13, line 2, with the following rewritten paragraph:

~~INITIALIZE_TABLE_VARS~~Initialize_table_vars step 260 sets all initial variables back to base values.

Please replace the paragraph beginning at page 13, line 3, with the following rewritten paragraph:

~~DETERMINE_OVER_THRESHOLD~~Determine_over_threshold step 254 determines if percent_used is over a threshold in step 255, if there are no more volumes in step 257, and if it is not GTS in step 259. Having made these determinations, the process proceeds to an error routine size step 261.

Please replace the paragraph beginning at page 13, line 7, with the following rewritten paragraph:

Alternatively, if total space is over a threshold, then the process proceeds to an ~~error routine space~~error_routine_space step 265.

Please replace the paragraph beginning at page 13, line 13, with the following rewritten paragraph:

In ~~**ERROR_ROUTINE_SPACE**~~error_routine_space step 265, a write potential error occurs. For ~~**ERROR_ROUTINE_SIZE**~~error_routine_size step 261, a write warning error occurs.

Please replace the paragraph beginning at page 13, line 15, with the following rewritten paragraph:

For ~~**FIND_LAST_VOLUME**~~find_last_volume step 275, it is determined if the last volume in the ~~listcat~~ LISTCAT is equal to the number passed from the ~~SUBROUTINE~~subroutine ~~listcat~~ LISTCAT in step 277. Then, the last_volume flag is set equal to that number in step 279.

Please replace the paragraph beginning at page 13, line 18, with the following rewritten paragraph:

For ~~**PROCESS_OSAM_GTS**~~process_osam_gts step 278, if a space_unit type is not in cylinders determination is made in step 280, then in step 281 current space is calculated into cylinder equivalent. A presently preferred approach to this calculation is based on secondary allocation. In this approach, it is determined how many cylinders are needed for 1 extent and for 2 extents in step 282. This is done to make the math processing a little quicker and to save a few lines of calculations. ~~SUBROUTINE~~Subroutine ~~subliste~~ SUBLISTC 300 is called in step 283.

~~SUBROUTINE~~ Subroutine ~~sub006~~ SUB006 500 is called in step 284. If it is GTS in step 285, then Process_GTS in step 286.

Please replace the paragraph beginning at page 14, line 1, with the following rewritten paragraph:

~~**PROCESS_GTS**~~Process_gts step 286 looks up volume information ~~Look~~look_up_vol_info in step 287. ~~**LOOK_UP_VOL_INFO**~~Look_up_vol_info step 287 proceeds to ~~Build~~build_new_sysin_card in step 288, calls ~~SUBROUTINE~~ subroutine IEHLISTR 600 in step 289, and calls ~~SUBROUTINE~~ subroutine SUBL002 700 in step 291. The above subroutines issue a dynamic IEHLIST that is used to determine how much free space is on a volume. In step 292, ~~Get~~ get total_free_cyls. Based on how much free space is left on the volume, one of the two following error routines will be called: ~~Write~~write_one_extent_error in step 294 or ~~Write~~write_two_extents_error in step 295.

Please replace the paragraph beginning at page 14, line 8, with the following rewritten paragraph:

~~**BUILD_NEW_SYSIN_CARD**~~Build_new_sysin_card step 288 formats a sysin card for an IEHLIST utility that will be dynamically executed. ~~**WRITE_ONE_EXTENT_ERROR**~~ Write_one_extent_error step 294 writes an output line. ~~**WRITE_TWO_EXTENTS_ERROR**~~ Write_two_extents_error step 295 writes an output line. ~~**WRITE_DATASET_NAME**~~

Write_dataset_name step 296 writes an output line. Finally, ~~WRITE_REPORT_ENDS~~ write_report_ends 297 writes output lines for the ending reports.

Please replace the paragraph beginning at page 14, line 24, with the following rewritten paragraph:

Figs. 4A-4C are a flow chart illustrating further details of the presently prefered subroutine SUB009 400. This subroutine 400 is a common module that will read the output from the subroutine SUBLISTC 300. After it reads the file, it will pass back to the calling module if the volumes are guaranteed space (GTS), how many volumes are used, and what these volumes are. The process 400 proceeds as follows once it has started at step 402 after it has been called. First, all variables are initialized in step 404. Then, a first record is read in step 406. A reformat recs step 408 is done. Next, variables are formatted in a format variables step 410. The reformatted variables are returned to the calling module in a return variables to calling module step 412. If a record being reformatted in step 408 has dataclass or volser indicators, data is accumulated as shown in Figs. 4B and 4C. In step 422, it is determined if a line has storage class data and it is determined in step 424 if it is guaranteed space or GTS. In step 425, storage class is analyzed to determine does it contain GTS and if yes, in step 427, the ~~gts~~ gts_flag is set to "YES". Returning to step 426, it is determined if a line has volser data. If yes, in step 428, the line is examined to see if it has "_ _ *" in it. If no, in step 430, one is added to the number of volumes and in step 432, the line is added to the volser. In step 434, the volser is put in the next available bracket. If at step 428 the answer is yes, then in step 429, one is added to the number of candidates.